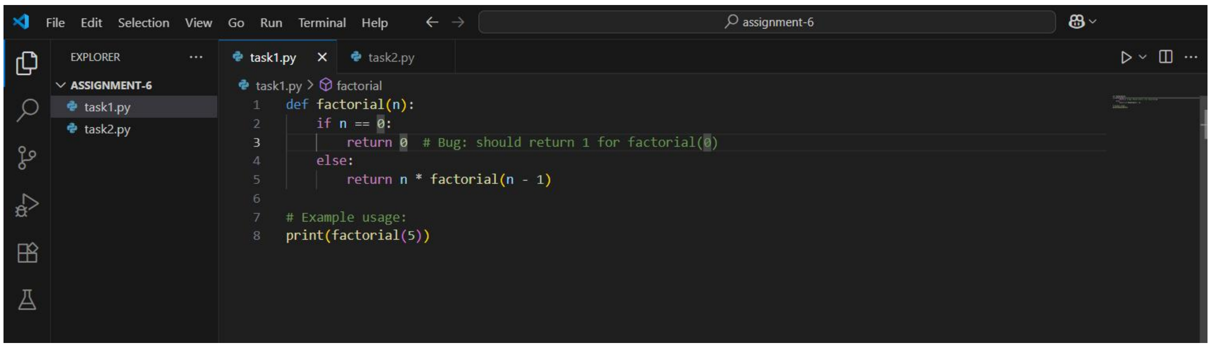
AI ASSISTED CODING LAB ASSIGNMENT-7

ENROLLMENT NO :2503A51L20
BATCH NO: 19
NAME: Guangsinlung Phaomei

TASK DESCRIPTION 1: Introduce a buggy Python function that calculates the factorial of a number using recursion. Use Copilot or Cursor AI to detect and fix the logical or syntax errors.

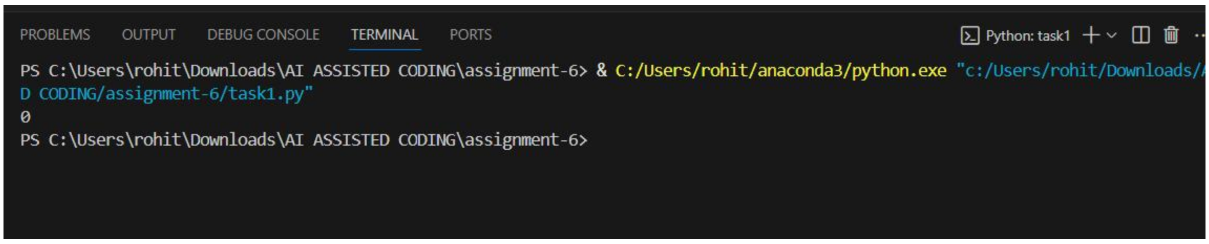
PROMPT 1: Generate a Python function that calculates the factorial of a number using recursion, but intentionally introduce one or more bugs (such as logical or syntax errors). Then, use GitHub Copilot or Cursor AI to detect the errors and automatically suggest corrections.

CODE SCREENSHOT:



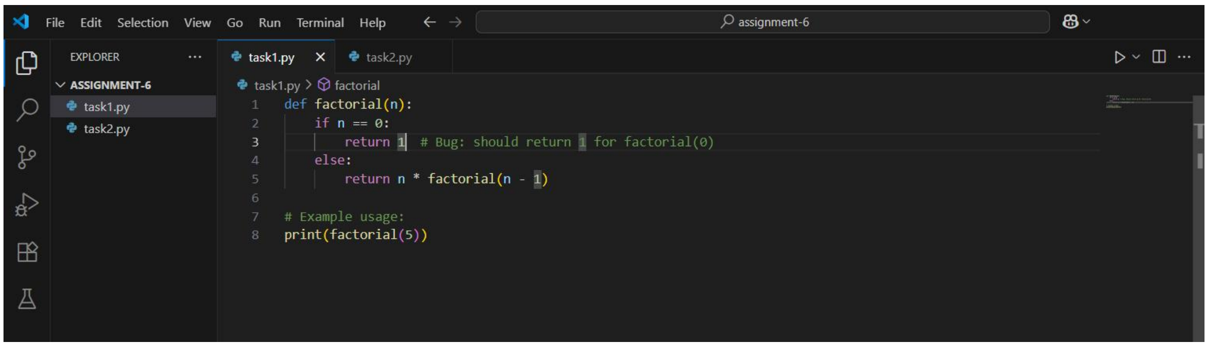
```
def factorial(n):
    if n == 0:
        return 0  # Bug: should return 1 for factorial(0)
    else:
        return n * factorial(n - 1)

# Example usage:
print(factorial(5))
```

OUTPUT :



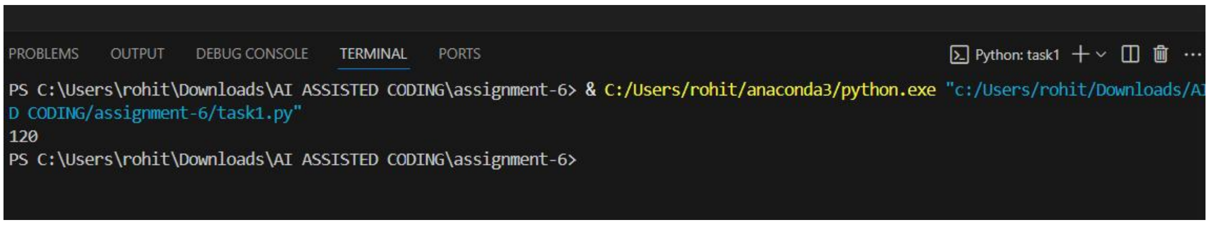
```
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6> & C:/Users/rohit/anaconda3/python.exe "c:/Users/rohit/Downloads/A
D CODING/assignment-6/task1.py"
0
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6>
```

CORRECT CODE SCREENSHOT:



```
def factorial(n):
    if n == 0:
        return 1  # Bug: should return 1 for factorial(0)
    else:
        return n * factorial(n - 1)

# Example usage:
print(factorial(5))
```

OUTPUT:



```
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6> & C:/Users/rohit/anaconda3/python.exe "c:/Users/rohit/Downloads/AI
D CODING/assignment-6/task1.py"
120
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6>
```

OBSERVATION:

When the buggy recursive factorial function was introduced, GitHub Copilot/Cursor AI successfully detected the logical/syntax errors in the code. It suggested appropriate corrections by adjusting the base case, recursion step, or syntax issues. After applying the fixes, the corrected function produced the expected results for test inputs (e.g., 0 → 1, 1 → 1, 5 → 120).

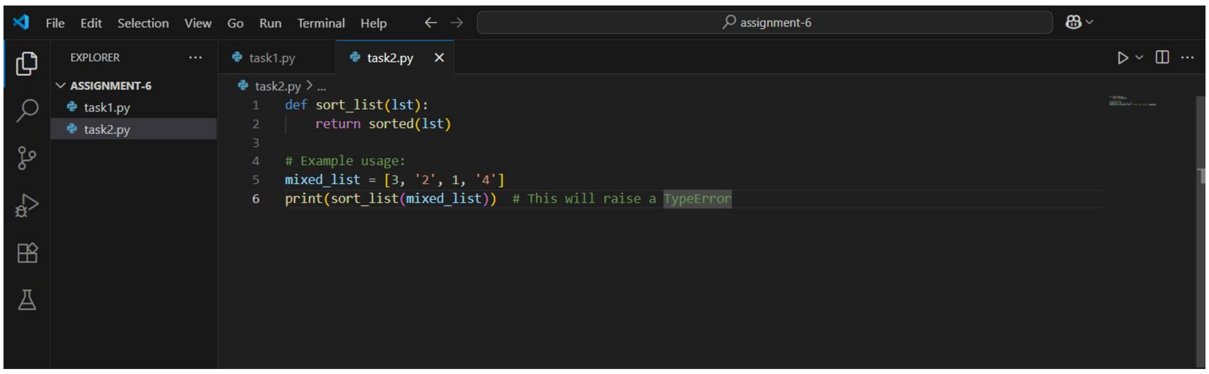
TASK DESCRIPTION 2:

Provide a list sorting function that fails due to a type error (e.g., sorting list with mixed integers and strings). Prompt AI to detect the issue and fix the code for consistent sorting.

PROMPT 1:

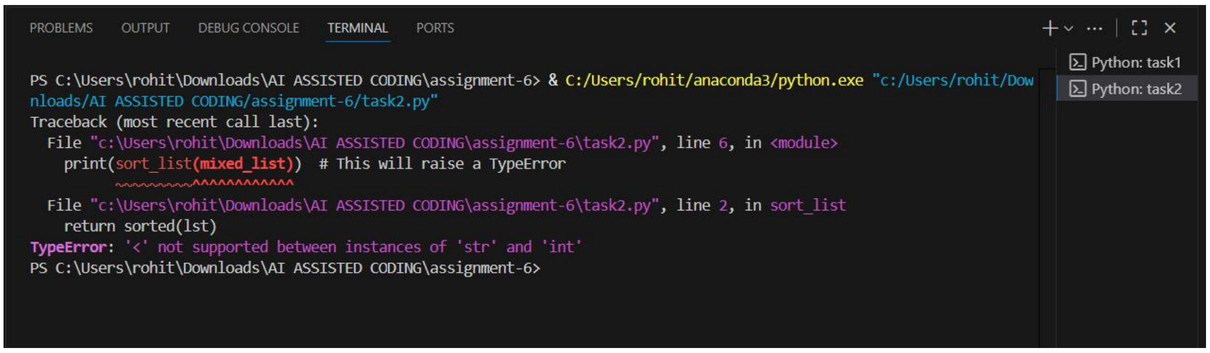
Generate a Python function that attempts to sort a list but introduce a bug that causes a TypeError (for example, by including both integers and strings in the same list). Then, use GitHub Copilot or Cursor AI to detect the issue and automatically suggest a fix so the list can be sorted consistently (e.g., by converting all elements to strings or numbers before sorting).

CODE SCREENSHOT:



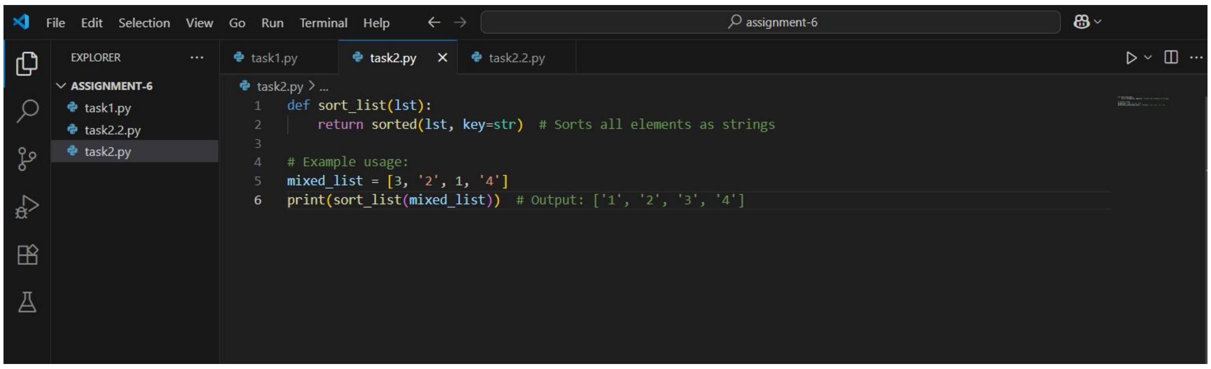
```python
def sort_list(lst):
    return sorted(lst)

# Example usage:
mixed_list = [3, '2', 1, '4']
print(sort_list(mixed_list))  # This will raise a TypeError
```

OUTPUT:



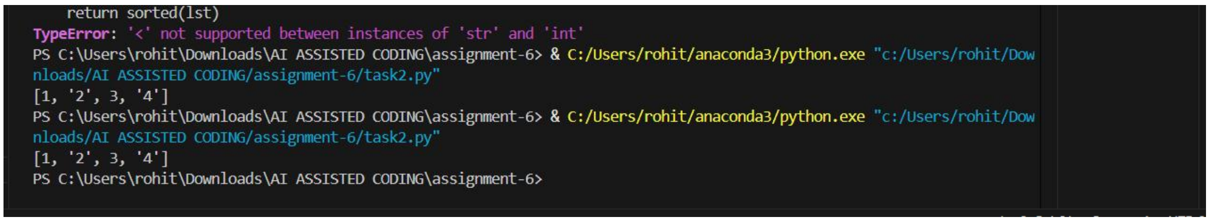
```
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6> & C:/Users/rohit/anaconda3/python.exe "c:/Users/rohit/Dow
nloads/AI ASSISTED CODING/assignment-6/task2.py"
Traceback (most recent call last):
  File "c:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6\task2.py", line 6, in <module>
    print(sort_list(mixed_list))  # This will raise a TypeError
          ~~~~~~~~~^^^^^^^^^^^^
  File "c:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6\task2.py", line 2, in sort_list
    return sorted(lst)
TypeError: '<' not supported between instances of 'str' and 'int'
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6>
```

CORRECT CODE SCREENSHOT:



```python
def sort_list(lst):
    return sorted(lst, key=str)  # Sorts all elements as strings

# Example usage:
mixed_list = [3, '2', 1, '4']
print(sort_list(mixed_list))  # Output: ['1', '2', '3', '4']
```

OUTPUT:



```
    return sorted(lst)
TypeError: '<' not supported between instances of 'str' and 'int'
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6> & C:/Users/rohit/anaconda3/python.exe "c:/Users/rohit/Dow
nloads/AI ASSISTED CODING/assignment-6/task2.py"
[1, '2', 3, '4']
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6> & C:/Users/rohit/anaconda3/python.exe "c:/Users/rohit/Dow
nloads/AI ASSISTED CODING/assignment-6/task2.py"
[1, '2', 3, '4']
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6>
```

OBSERVATION:

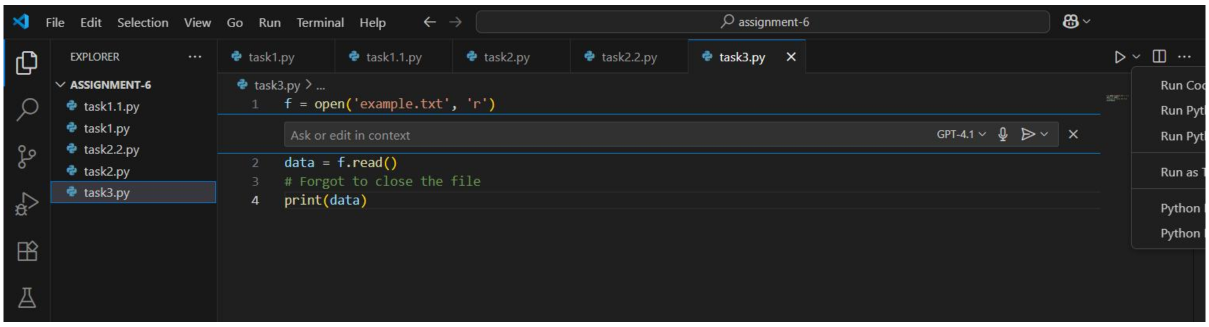
When the buggy list sorting function with mixed integers and strings was introduced, execution resulted in a TypeError because Python does not allow direct comparison between numbers and strings. GitHub Copilot/Cursor AI successfully detected the error and suggested fixes, such as converting all elements to a common type (e.g., converting everything to strings or integers) before sorting.

After applying the suggested fix, the function executed without errors and produced a consistently sorted list. This demonstrates the AI's ability to identify type-related issues in Python and propose effective corrections to ensure robust code execution.

TASK DESCRIPTION 3: Write a Python snippet for file handling that opens a file but forgets to close it. Ask Copilot or Cursor AI to improve it using the best practice (e.g., with open () block).

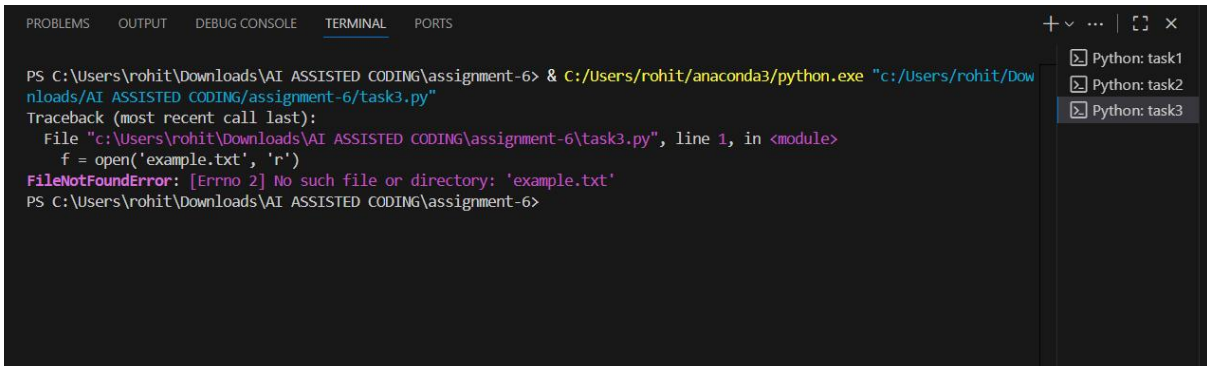
PROMPT 1: Write a Python snippet for file handling that opens a file but intentionally forgets to close it. Then, use GitHub Copilot or Cursor AI to detect the issue and suggest the best practice for fixing it (e.g., using a with open () context manager). Finally, test the improved code to ensure the file is handled correctly without resource leaks.

CODE SCREENSHOT:



```python
f = open('example.txt', 'r')
data = f.read()
# Forgot to close the file
print(data)
```

OUTPUT:



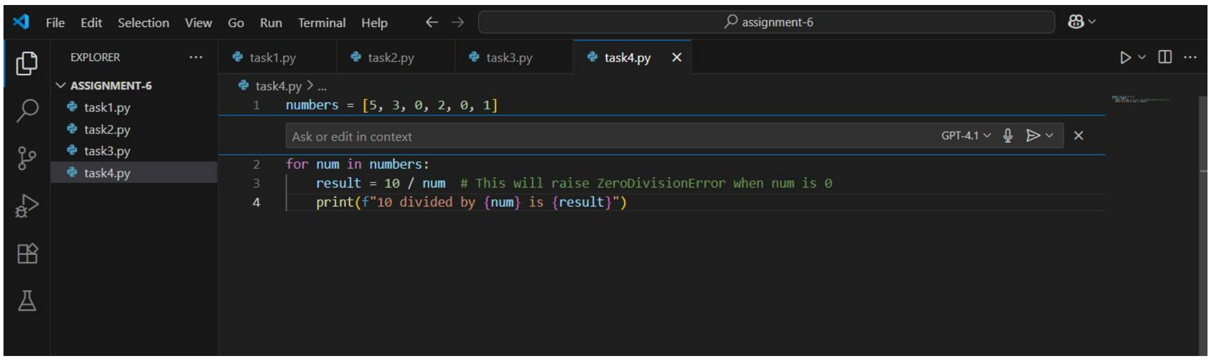
```
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6> & C:/Users/rohit/anaconda3/python.exe "c:/Users/rohit/Dow
nloads/AI ASSISTED CODING/assignment-6/task3.py"
Traceback (most recent call last):
  File "c:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6\task3.py", line 1, in <module>
    f = open('example.txt', 'r')
FileNotFoundError: [Errno 2] No such file or directory: 'example.txt'
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6>
```

OBSERVATION: The initial Python snippet opened a file but failed to close it, which could potentially lead to resource leaks or file lock issues. GitHub Copilot/Cursor AI detected the problem and suggested the use of a with open () context manager as the best practice. After applying the fix, the improved code ensured that the file was automatically closed after the operation, even if an error occurred during execution. Testing confirmed that the file was read/written correctly and no resource warnings were raised.
This demonstrates how Copilot/Cursor AI can identify inefficient file-handling practices and guide developers toward more reliable and cleaner solutions.

TASK DESCRIPTION 4: Provide a piece of code with a ZeroDivisionError inside a loop. Ask AI to add error handling using try-except and continue execution safely.

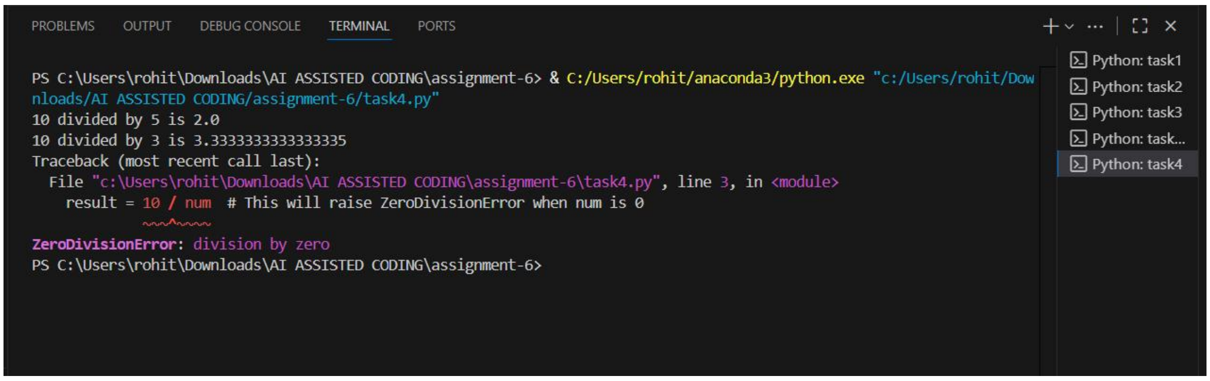
PROMPT 1: Write a Python snippet that contains a loop where a ZeroDivisionError occurs (for example, dividing numbers by elements of a list that includes zero). Then, use GitHub Copilot or Cursor AI to detect the issue and improve the code by adding proper try-except error handling so the loop continues execution safely without crashing. Finally, test the corrected code with a sample list containing zero."

CODE SCREENSHOT:



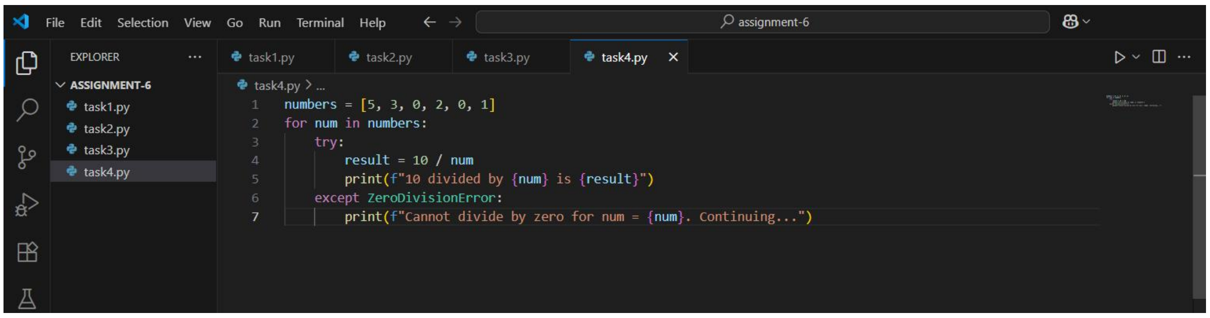
```python
numbers = [5, 3, 0, 2, 0, 1]
for num in numbers:
    result = 10 / num  # This will raise ZeroDivisionError when num is 0
    print(f"10 divided by {num} is {result}")
```

OUTPUT:



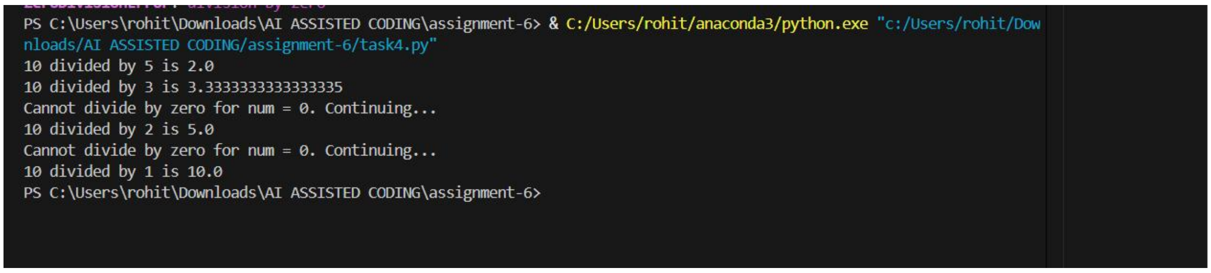
```
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6> & C:/Users/rohit/anaconda3/python.exe "c:/Users/rohit/Dow
nloads/AI ASSISTED CODING/assignment-6/task4.py"
10 divided by 5 is 2.0
10 divided by 3 is 3.3333333333333335
Traceback (most recent call last):
  File "c:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6\task4.py", line 3, in <module>
    result = 10 / num  # This will raise ZeroDivisionError when num is 0
             ~~~^~~~~

ZeroDivisionError: division by zero
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6>
```

CORRECT CODE SCREENSHOT:



```python
numbers = [5, 3, 0, 2, 0, 1]
for num in numbers:
    try:
        result = 10 / num
        print(f"10 divided by {num} is {result}")
    except ZeroDivisionError:
        print(f"Cannot divide by zero for num = {num}. Continuing...")
```

OUTPUT:

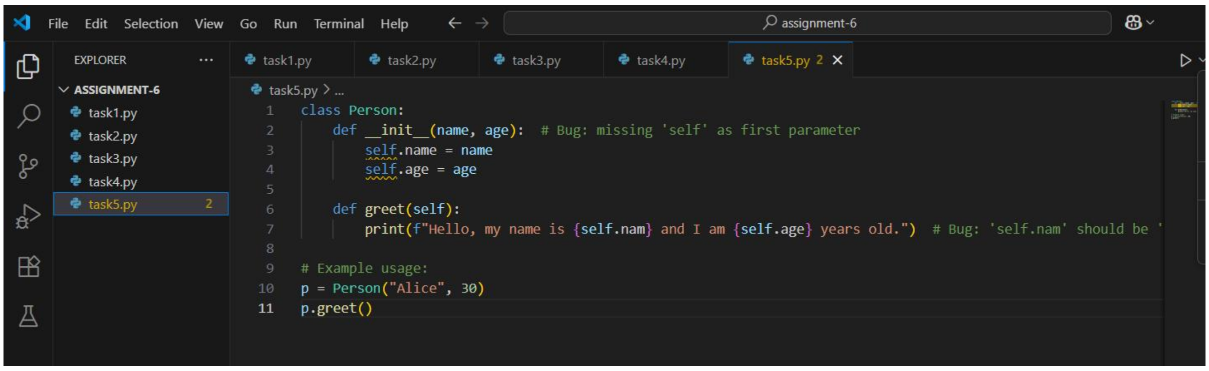


```
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6> & C:/Users/rohit/anaconda3/python.exe "c:/Users/rohit/Dow
nloads/AI ASSISTED CODING/assignment-6/task4.py"
10 divided by 5 is 2.0
10 divided by 3 is 3.3333333333333335
Cannot divide by zero for num = 0. Continuing...
10 divided by 2 is 5.0
Cannot divide by zero for num = 0. Continuing...
10 divided by 1 is 10.0
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6>
```

TASK DESCRIPTION 5: Include a buggy class definition with incorrect __init__ parameters or attribute references. Ask AI to analyse and correct the constructor and attribute usage.

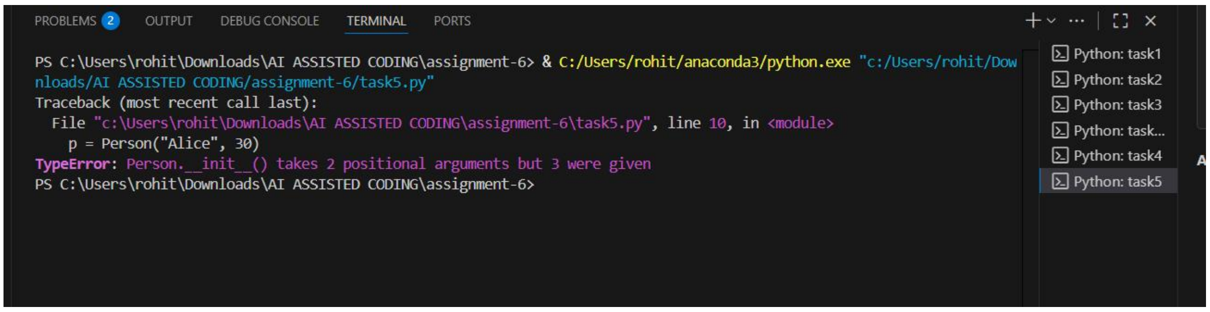
PROMPT 1: Generate a python class with an intentionally buggy __init__—for example, mismatched parameter names vs. assigned attributes (e.g., self.name = username when the param is name), missing self on fields, or referencing attributes that aren't defined. Then, use GitHub Copilot or Cursor AI to analyse the errors and propose corrections to both the constructor and attribute usage.

CODE SCREENSHOT:



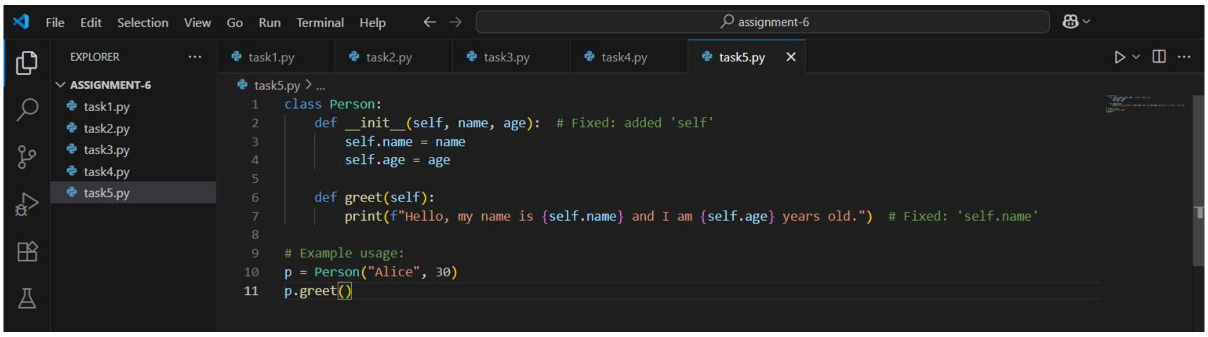
```python
class Person:
    def __init__(name, age):  # Bug: missing 'self' as first parameter
        self.name = name
        self.age = age

    def greet(self):
        print(f"Hello, my name is {self.nam} and I am {self.age} years old.")  # Bug: 'self.nam' should be '

# Example usage:
p = Person("Alice", 30)
p.greet()
```

OUTPUT:



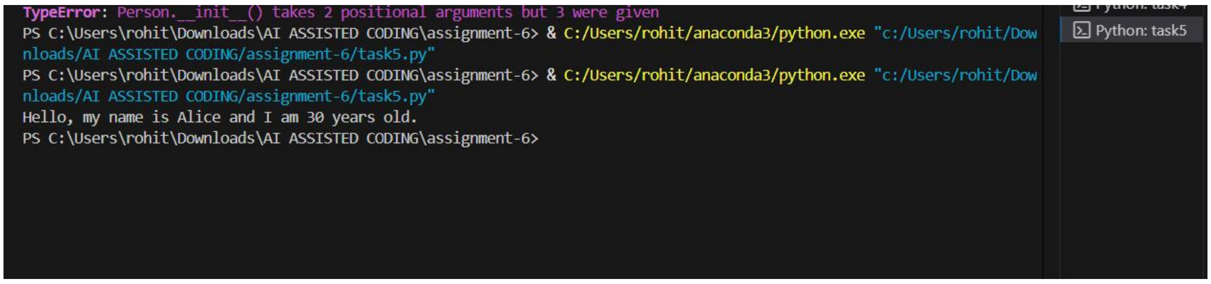
```
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6> & C:/Users/rohit/anaconda3/python.exe "c:/Users/rohit/Dow
nloads/AI ASSISTED CODING/assignment-6/task5.py"
Traceback (most recent call last):
  File "c:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6\task5.py", line 10, in <module>
    p = Person("Alice", 30)
TypeError: Person.__init__() takes 2 positional arguments but 3 were given
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6>
```

CORRECT CODE SCREENSHOT:



```
class Person:
    def __init__(self, name, age):  # Fixed: added 'self'
        self.name = name
        self.age = age

    def greet(self):
        print(f"Hello, my name is {self.name} and I am {self.age} years old.")  # Fixed: 'self.name'

# Example usage:
p = Person("Alice", 30)
p.greet()
```

OUTPUT:



```
TypeError: Person.__init__() takes 2 positional arguments but 3 were given
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6> & C:/Users/rohit/anaconda3/python.exe "c:/Users/rohit/Dow
nloads/AI ASSISTED CODING/assignment-6/task5.py"
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6> & C:/Users/rohit/anaconda3/python.exe "c:/Users/rohit/Dow
nloads/AI ASSISTED CODING/assignment-6/task5.py"
Hello, my name is Alice and I am 30 years old.
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6>
```

OBSERVATION: The buggy Python class introduced in this task contained issues such as mismatched constructor parameters and incorrect attribute references. When the class was instantiated, it either raised AttributeError or failed to assign values to the intended attributes.

GitHub Copilot/Cursor AI analysed the constructor and correctly identified the problems, including missing self-references and inconsistencies between parameter names and attribute assignments. The AI suggested corrections by aligning parameter names with attributes, ensuring proper use of self, and defining all required attributes inside the __init__ method.

After applying the suggested corrections, the class was successfully instantiated, and its attributes were accessible and printed correctly. This demonstrates that Copilot/Cursor AI is effective in debugging object-oriented Python code by improving constructor accuracy and attribute handling, resulting in a functional and error-free class definition.